浙江大學
ZHEJIANG UNIVERSITY

Computer Architecture Experiment

Topic 6. Pipelined CPU supporting Interrupt

浙江大学计算机学院
陈文智
chenwz@zju.edu.cn

实验操作流程

- 阅读实验文档，理解CP0的工作原理和中断相关机制的实现方式
- 以前一次实验为基础，增加协处理器CP0，支持中断指令及中断相关机制
- 对处理器进行仿真，检验处理器的仿真结果是否符合要求。
- 综合工程并下载至开发板，在单步执行的过程中检查调试屏幕的输出，检验处理器的执行过程是否正确。

实验验收标准

- 仿真执行过程中，处理器的行为和内部控制信号均符合要求。
- 下载至开发板后的单步执行过程中，寄存器的变化过程和最终执行结果与测试程序相吻合。

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures
- Precaution
- Checkpoints

Experiment Purpose

- Understand the principle of CPU Interrupt and its processing procedure.
- Understand the function of CP0 coprocessor.
- Master the design methods of pipelined CPU supporting interrupt.
- master methods of program verification of Pipelined CPU supporting interrupt.

Experiment Task

- **Design of Pipelined CPU supporting Interrupt.**
  - Design CP0
  - Design CPU Controller
  - Design datapath

- **Verify the Pipelined CPU with program and observe the execution of program**

What jobs does CP0 do?

- **CPU configuration**
- **Cache control**
- **Exception/interrupt control**
- **Memory management unit control**
- **Miscellaneous**

Exception/interrupt control

- CP Instructions
- Simple interrupt control
- Some Mechanisms
- Simple CP0 design
- Code Example

CP Interrupt Instructions (1)

- **Register index bits: 5 bits.**
- **MIPS32 uses another 3 bits “SEL” to extend index bits**
- **CP0 Interrupt Instructions**
  - MTC0
  - MFC0
  - ERET

CP Interrupt Instructions (2)

- **Format: MFC0 rt, rd, sel**
- **Function: GPR[rt] = CPR[0, rd, sel]**

31 26	25 21	20 16	15 11	10 3	2 0
COP0 010000	MF 00000	RT	RD	0 0000 0000	SEL

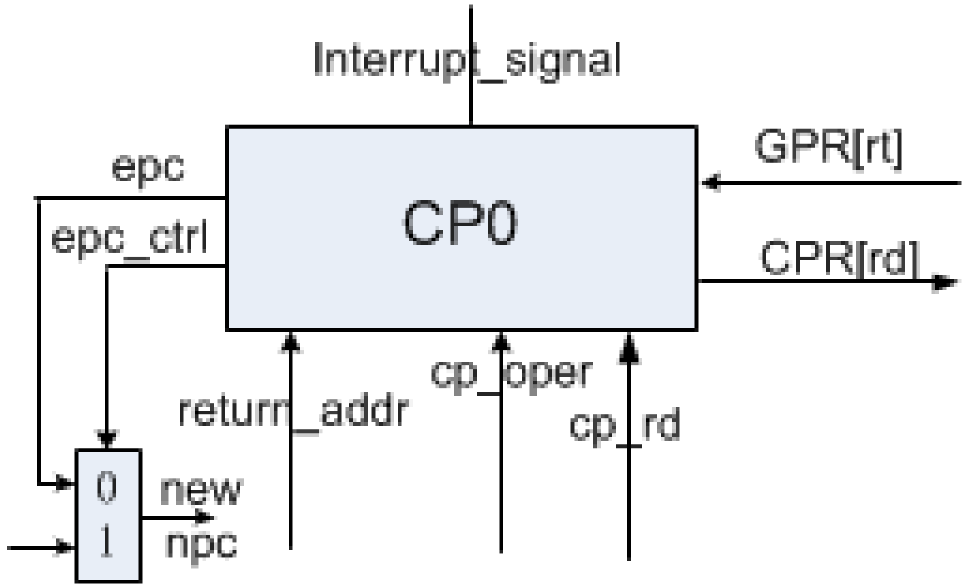
- **Format: MTC0 rt, rd, sel**
- **Function: CPR[0, rd, sel] = GPR[rt]**

31 26	25 21	20 16	15 11	10 3	2 0
COP0 010000	MT 00100	RT	RD	0 0000 0000	SEL

- **ERET**

31 26	25	24 6	5 0
COP0 010000	CO 1	0 000 0000 0000 0000 0000	ERET 011000

Simple interrupt control (1)



MFC0 rt, rd, sel **#GPR[rt] = CPR[0, rd, sel]**

MTC0 rt, rd, sel **#CPR[0, rd, sel] = GPR[rt]**

Simple interrupt control (2)

- **Interrupt is detected**
  - Instructions in IF is killed (ID_RST= 1)
  - Return address (Delay slot)

- **Interrupt returns**
  - Modify npc

Some Mechanisms

- **Interrupt signal should be saved**
- **Interrupt handler could not be re-entered**

Simple CP0 design

- **Interrupt register file**
  - Exception Handler Base Register (EHBR)
  - Exception Program Counter Register (EPCR)
- **MTC0: Write Interrupt register file**
- **MFC0: Read Interrupt register file**
- **Interrupt is detected: jump to EHBR (External signal)**
- **ERET: jump to EPCR (From CPU controller)**

Code Example (1)

```
module cp0 (
        input wire clk,  // main clock
        // debug
        `ifdef DEBUG
        input wire [4:0] debug_addr,  // debug address
        output reg [31:0] debug_data,  // debug data
        `endif
        // operations (read in ID stage and write in EXE stage)
        input wire [1:0] oper,  // CP0 operation type
        input wire [4:0] addr_r,  // read address
        output reg [31:0] data_r,  // read data
        input wire [4:0] addr_w,  // write address
        input wire [31:0] data_w,  // write data
        // exceptions (check exceptions in MEM stage)
        input wire rst,  // synchronous reset
        input wire ir_en,  // interrupt enable
        input wire ir_in,  // external interrupt input
        input wire [31:0] ret_addr,  // target instruction address to store when interrupt occurred
        output reg jump_en,  // force jump enable signal when interrupt authorised or ERET occurred
        output reg [31:0] jump_addr  // target instruction address to jump to
        );
```

Code Example (2)

```
// interrupt determination
wire ir;
reg ir_wait = 0, ir_valid = 1;
reg eret = 0;
always @(posedge clk) begin
        if (rst)
                ir_wait <= 0;
        else if (ir_in)
                ir_wait <= 1;
        else if (eret)
                ir_wait <= 0;
end
always @(posedge clk) begin
        if (rst)
                ir_valid <= 1;
        else if (eret)
                ir_valid <= 1;
        else if (ir)
                ir_valid <= 0;  // prevent exception reenter
end
assign ir = ir_en & ir_wait & ir_valid;
```

Code Example (3)

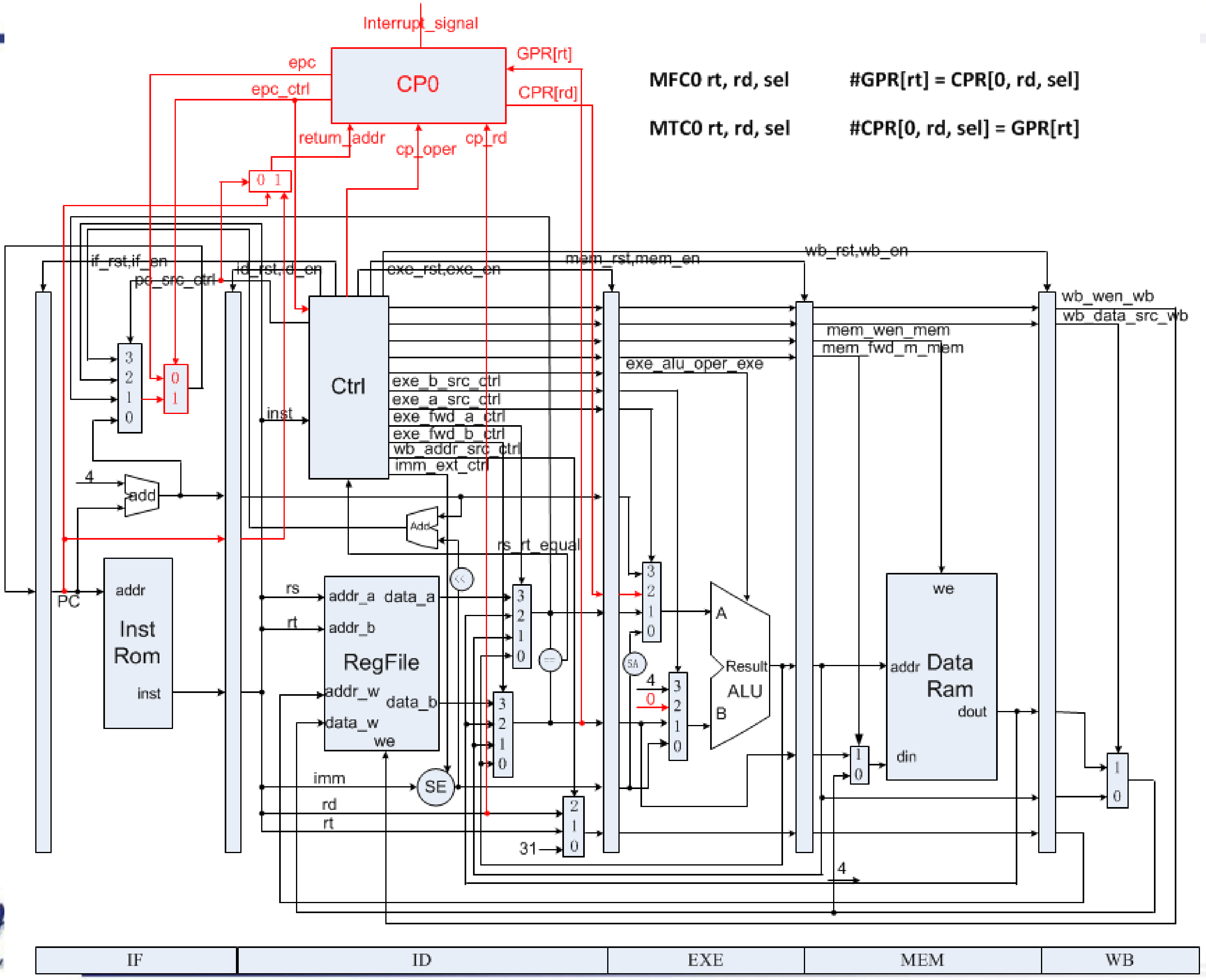
```
// Exception Handler Base Register
always @(posedge clk) begin
        ……
end

// Exception Program Counter Register
always @(posedge clk) begin
        ……
end

// jump determination
always @(*) begin
        ……
end
```

Code Example (4)

```
// CP0 registers
localparam
        //CP0_SR    = 0,
        //CP0_EAR   = 1,
        CP0_EPCR  = 2,
        CP0_EHBR  = 3;
        //CP0_IER   = 4,
        //CP0_ICR   = 5,
        //CP0_PDBR  = 6,
        //CP0_TIR   = 7,
        //CP0_WDR   = 8;
```

```
// EXE CP operations
localparam
        EXE_CP_NONE   = 0,
        EXE_CP_STORE  = 1,
        EXE_CP0_ERET  = 2;
```

Datapath of CPU supporting interrupt.



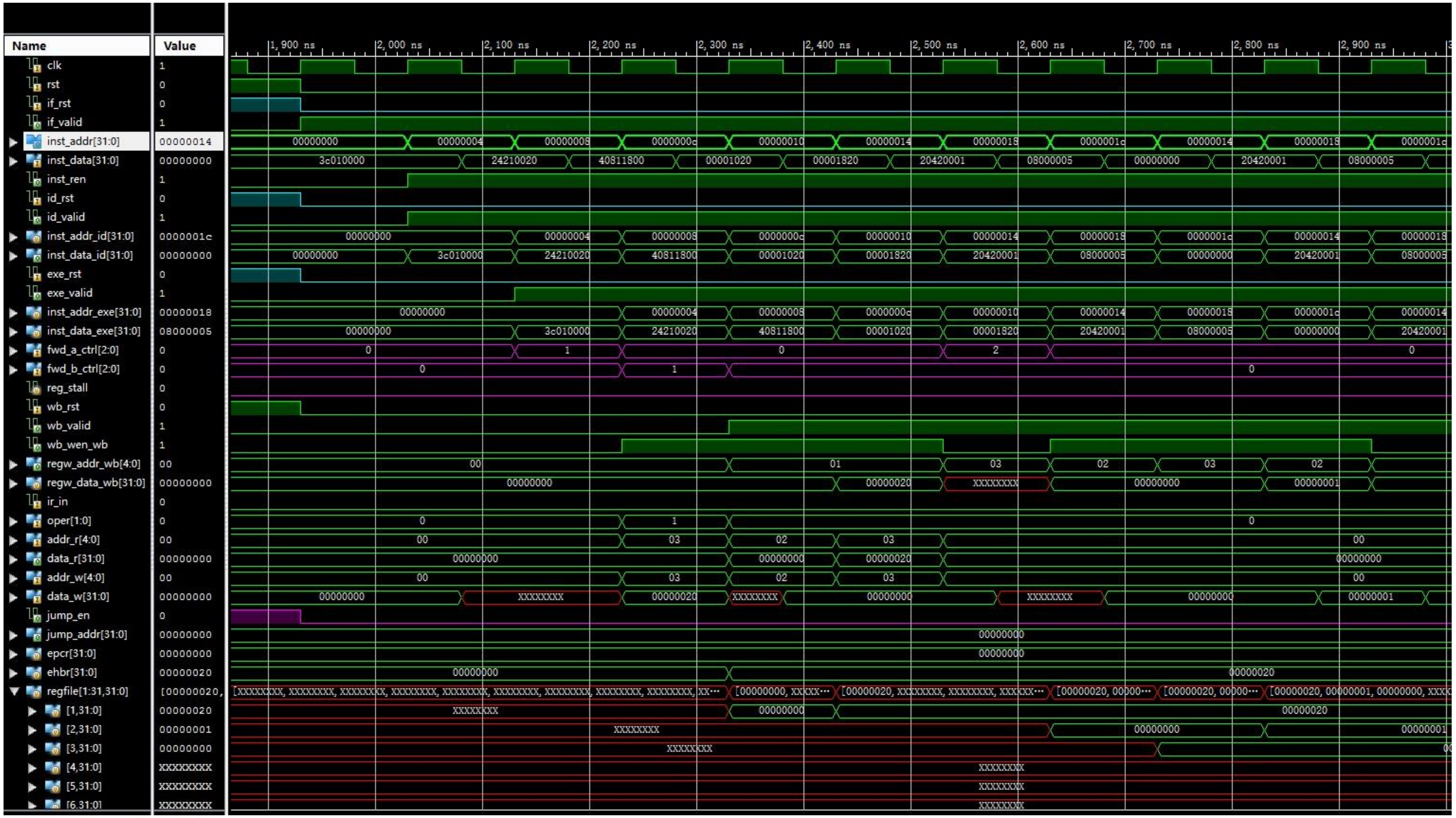
MFC0 rt, rd, sel **#GPR[rt] = CPR[0, rd, sel]**

MTC0 rt, rd, sel **#CPR[0, rd, sel] = GPR[rt]**

Instr. Mem.(3)

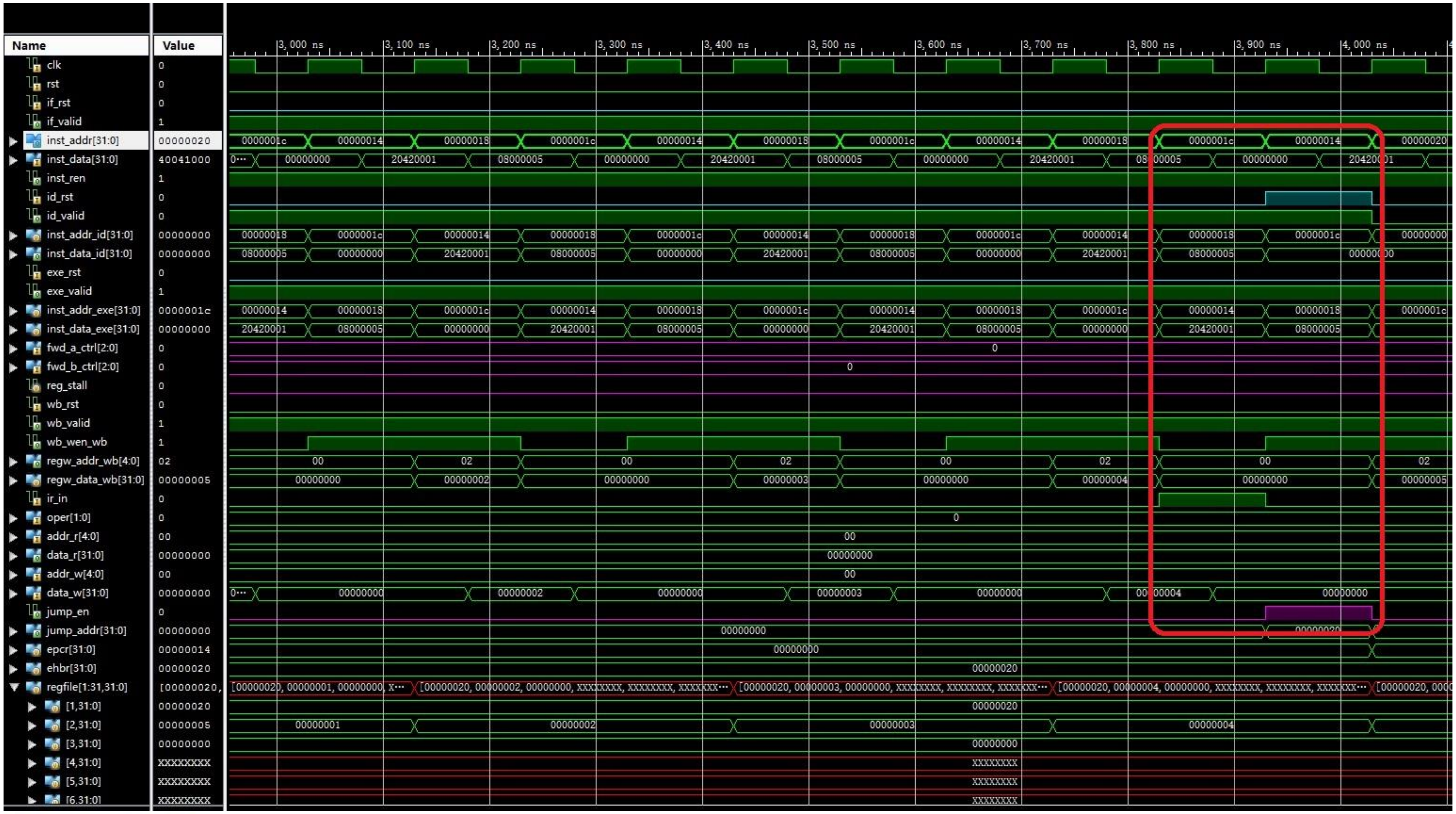
```
0:    3c010000    lui     R1,0x0       //main entry
4:    24210020    addiu   R1,R1,32
8:    40811800    mtc0    R1, R3
c:    00001020    add     R2,R0,R0
10:   00001820    add     R3,R0,R0
14:   20420001    addi    R2,R2,1      //loop
18:   08000005    j       14
1c:   00000000    nop
20:   40041000    mfc0    R4,R2        //handler
24:   20630001    addi    R3,R3,1
28:   42000018    eret
2c:   00000000    nop
```

Simulation (1)



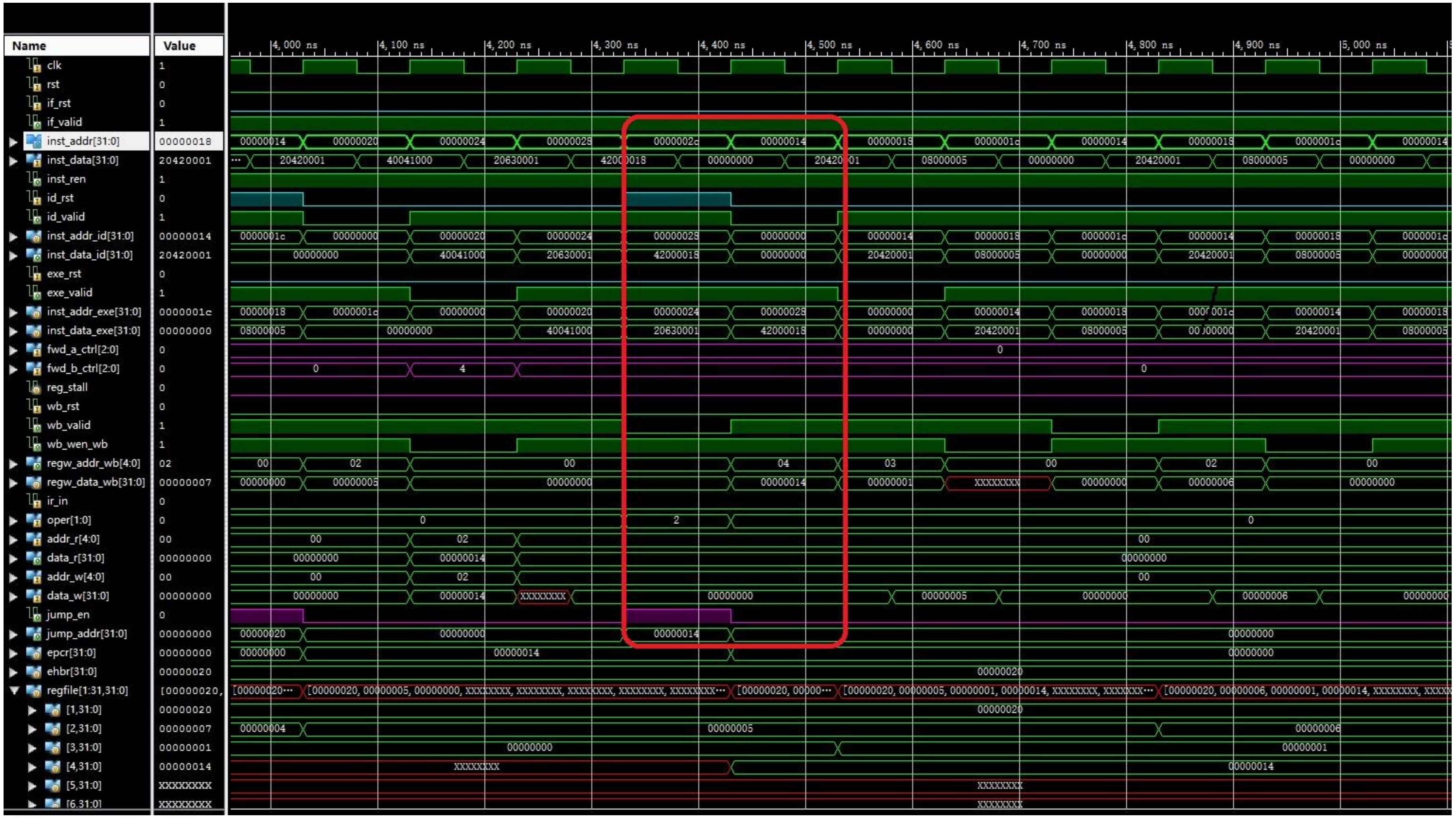
Name	Value
clk	1
rst	0
if_rst	0
if_valid	1
inst_addr[31:0]	00000014
inst_data[31:0]	00000000
inst_ren	1
id_rst	0
id_valid	1
inst_addr_id[31:0]	0000001c
inst_data_id[31:0]	00000000
exe_rst	0
exe_valid	1
inst_addr_exe[31:0]	00000018
inst_data_exe[31:0]	08000005
fwd_a_ctrl[2:0]	0
fwd_b_ctrl[2:0]	0
reg_stall	0
wb_rst	0
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	00
regw_data_wb[31:0]	00000000
ir_in	0
oper[1:0]	0
addr_r[4:0]	00
data_r[31:0]	00000000
addr_w[4:0]	00
data_w[31:0]	00000000
jump_en	0
jump_addr[31:0]	00000000
epcr[31:0]	00000000
ehbr[31:0]	00000020
regfile[1:31,31:0]	[00000020,
[1,31:0]	00000020
[2,31:0]	00000001
[3,31:0]	00000000
[4,31:0]	XXXXXXXX
[5,31:0]	XXXXXXXX
[6,31:0]	XXXXXXXX

Simulation (2)



Name	Value
clk	0
rst	0
if_rst	0
if_valid	1
inst_addr[31:0]	00000020
inst_data[31:0]	40041000
inst_ren	1
id_rst	0
id_valid	0
inst_addr_id[31:0]	00000000
inst_data_id[31:0]	00000000
exe_rst	0
exe_valid	1
inst_addr_exe[31:0]	0000001c
inst_data_exe[31:0]	00000000
fwd_a_ctrl[2:0]	0
fwd_b_ctrl[2:0]	0
reg_stall	0
wb_rst	0
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	02
regw_data_wb[31:0]	00000005
ir_in	0
oper[1:0]	0
addr_r[4:0]	00
data_r[31:0]	00000000
addr_w[4:0]	00
data_w[31:0]	00000000
jump_en	0
jump_addr[31:0]	00000000
epcr[31:0]	00000014
ehbr[31:0]	00000020
regfile[1:31,31:0]	[00000020,
[1,31:0]	00000020
[2,31:0]	00000005
[3,31:0]	00000000
[4,31:0]	XXXXXXXX
[5,31:0]	XXXXXXXX
[6,31:0]	XXXXXXXX

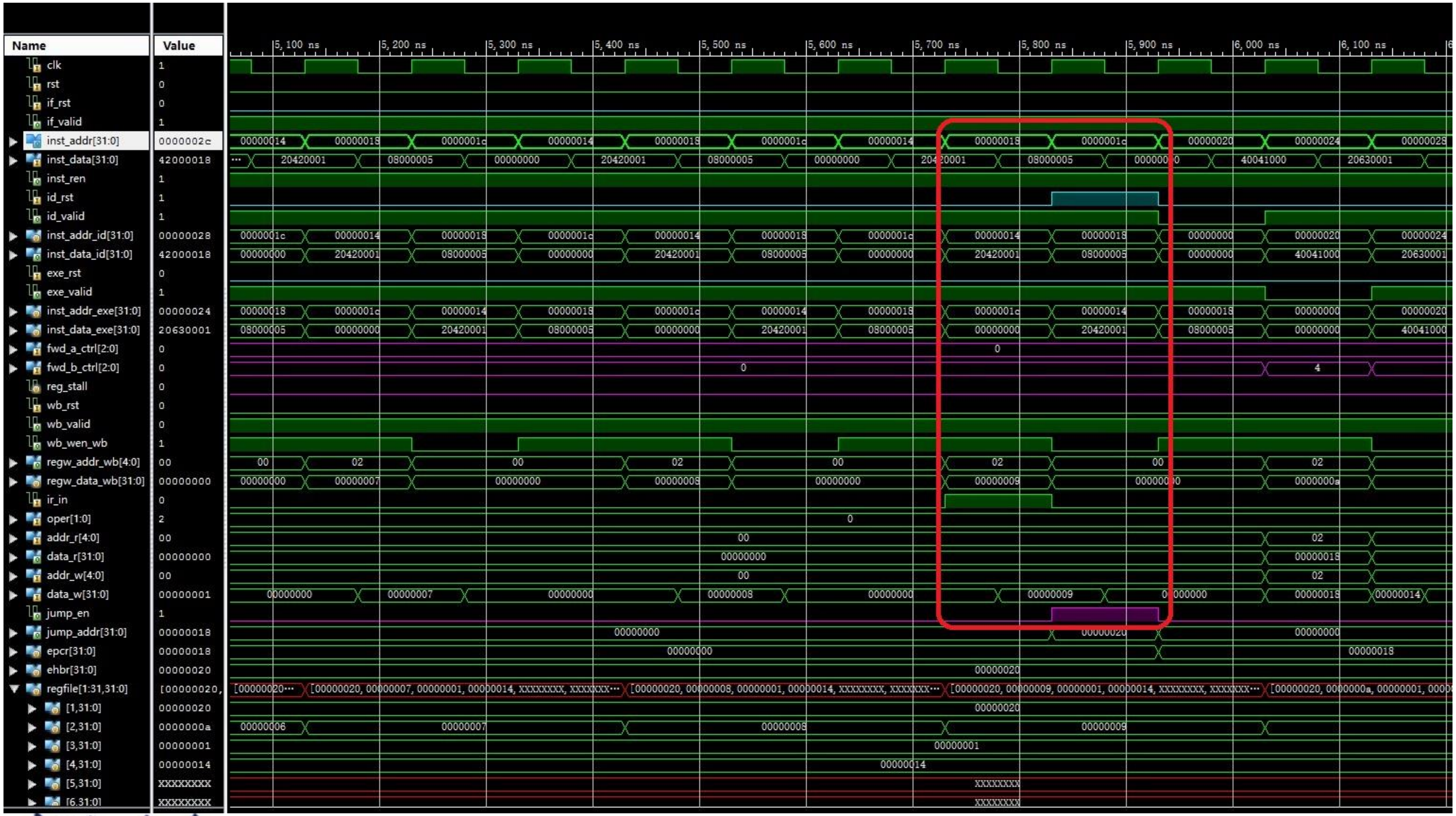
Simulation (3)



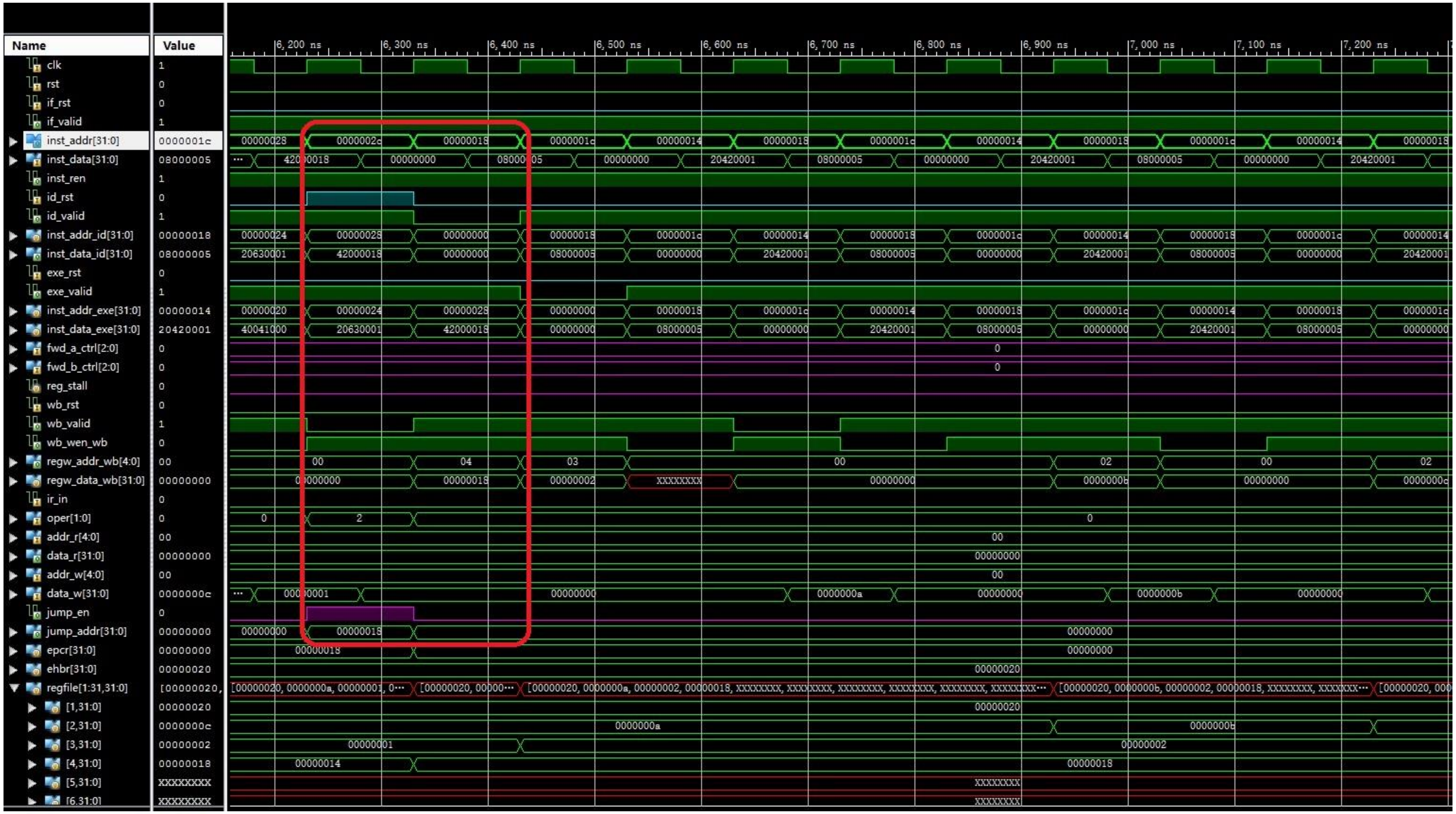
Name	Value
clk	1
rst	0
if_rst	0
if_valid	1
inst_addr[31:0]	00000018
inst_data[31:0]	20420001
inst_ren	1
id_rst	0
id_valid	1
inst_addr_id[31:0]	00000014
inst_data_id[31:0]	20420001
exe_rst	0
exe_valid	1
inst_addr_exe[31:0]	0000001c
inst_data_exe[31:0]	00000000
fwd_a_ctrl[2:0]	0
fwd_b_ctrl[2:0]	0
reg_stall	0
wb_rst	0
wb_valid	1
wb_wen_wb	1
regw_addr_wb[4:0]	02
regw_data_wb[31:0]	00000007
ir_in	0
oper[1:0]	0
addr_r[4:0]	00
data_r[31:0]	00000000
addr_w[4:0]	00
data_w[31:0]	00000000
jump_en	0
jump_addr[31:0]	00000000
epcr[31:0]	00000000
ehbr[31:0]	00000020
regfile[1:31,31:0]	[00000020,
[1,31:0]	00000020
[2,31:0]	00000007
[3,31:0]	00000001
[4,31:0]	00000014
[5,31:0]	XXXXXXXX
[6,31:0]	XXXXXXXX

浙江大学 ZheJiang University

Simulation (4)



Simulation (5)



Name	Value
clk	1
rst	0
if_rst	0
if_valid	1
inst_addr[31:0]	0000001c
inst_data[31:0]	08000005
inst_ren	1
id_rst	0
id_valid	1
inst_addr_id[31:0]	00000018
inst_data_id[31:0]	08000005
exe_rst	0
exe_valid	1
inst_addr_exe[31:0]	00000014
inst_data_exe[31:0]	20420001
fwd_a_ctrl[2:0]	0
fwd_b_ctrl[2:0]	0
reg_stall	0
wb_rst	0
wb_valid	1
wb_wen_wb	0
regw_addr_wb[4:0]	00
regw_data_wb[31:0]	00000000
ir_in	0
oper[1:0]	0
addr_r[4:0]	00
data_r[31:0]	00000000
addr_w[4:0]	00
data_w[31:0]	0000000c
jump_en	0
jump_addr[31:0]	00000000
epcr[31:0]	00000000
ehbr[31:0]	00000020
regfile[1:31,31:0]	[00000020,
[1,31:0]	00000020
[2,31:0]	0000000c
[3,31:0]	00000002
[4,31:0]	00000018
[5,31:0]	xxxxxxxx
[6,31:0]	xxxxxxxx

FPGA

```
// interrupt
wire interrupt;
reg interrupt_prev;
always @(posedge clk) begin
        interrupt_prev <= interrupter;
end
assign interrupt = ~interrupt_prev & interrupter;
```

- **Interrupter： West Button**
- **Interrupt is triggered at posedge of West Button’s Pushing**

Checkpoints

- **CP 1:**

  Waveform Simulation of the Pipelined CPU with the verification program

- **CP 2:**

  FPGA Implementation of the Pipelined CPU with the verification program

Thanks!

浙江大學
ZheJiang University